TomcatDataSourceRealms

Tomcat DataSource Realms

Introduction

Useful information for configuring Tomcat DataSourceRealms is spread out over three documents. This often leads to people using a simple JDBCRealm for authentication and authorization. There are at least two challenges when using a simple JDBCRealm for this purpose.

1. High degree of synchronization can lead to poor performance on high volume sites
2. Database connection timeouts on low volume sites can lead authentication failure

The Tomcat DataSource realm addresses these issues by using a JNDI datasource. The realm can be configured with adequate pooling parameters to reduce synchronization issues, and a validationQuery to prevent database connection timeouts.

This document takes information from three Tomcat documents to describe some ways to configure DataSource Realms for authorization and authentication.

Environments

The following two environments are used while writing this document.

Component	Version
OS	Fedora 13 32 bit
JDK/JRE	1.6.0_20
Tomcat	6.0.26
Apache Derby	10.5.3.0
IDE	NetBeans 6.8
Hibernate	3.2.5 ga
JSF	1.2
OS	Windows/XP Professional SP 4 32 bit
JDK/JRE	1.6.0_20
Tomcat	6.0.26
Apache Derby	5.1.31
IDE	NetBeans 6.8
Hibernate	10.5.3.0
JSF	1.2

In order to test these configurations, a simple JSF / Hibernate / CRUD application based on the the following NetBeans JSF Tutorial was used. The entire application was wrapped up in a BASIC authentication scheme using the following web.xml portion.

```
<security-constraint>
    <display-name>Entire application</display-name>
    <web-resource-collection>
        <web-resource-name>Everything</web-resource-name>
        <description>coarse grained approach</description>
        <url-pattern>/faces/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description/>
        <role-name>user</role-name>
    </auth-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Hibernate Application</realm-name>
</login-config>
<security-role>
    <description>generic user of application</description>
    <role-name>user</role-name>
</security-role>
```

Finally, a separate authorization and authentication database was created in Apache Derby. This database consists of three tables and one view.

1. USERS table with USERNAME and PASSWORD columns
2. ROLES table with ROLENAME and DESCRIPTION columns
3. ASSIGN table
    a. foreign keys for USERNAME and ROLENAME from the USERS and ROLES tables
    b. STATE column for active and inactive assignments
4. AUTH view with USERNAME and ROLENAME
    a. joined on USERS and USERNAME
    b. joined on ROLES and ROLENAME
    c. conditional on ASSIGN.STATE being active

Overview

The DataSource realm actually consists of two components.

The first component is a JNDI JDBC data source resource. Documentation for setting this up can be found here:

- JDBC Data Sources

IMPORTANT NOTE

Using a JNDI JDBC data source resource requires the JDBC driver to be visible to Tomcat. Thus, the JDBC driver needs to be placed in $CATALINA_BASE/lib (for Tomcat 6). Once this is done, **do not put this driver in the application's WEB-INF/lib directory**.

The second component is the actual Realm. Documentation for setting this up can be found in the following locations:

- Realm Component
- DataSource Realm

In particular, pay attention to the table and column mappngs required for the DataSource realm. Combining this information leads to a working DataSource realm for authentication and authorization.

Three Scenarios

Three configuration scenarios are presented below.

- Everything in META-INF/context.xml which provides an application - specific configuration
- GlobalNamingResources and META-INF/context.xml which provides for multiple applications selectively using authentication
- Everything in $CATALINA_BASE/conf/server.xml which provides a global Host or Engine configuration

Everything in META-INF/context.xml

This is appropriate when each web application might use different authentication and authorization databases. This is also the simplest configuration to manage, since all configuration elements are in one location. Finally, this configuration will produce the most portable (between Tomcat installations) war file.

Resource Element

A Resource element is created in META-INF/context.xml to describe the database connection and provide a JNDI name. This is the same type of Resource description that is used for application - level JNDI data source. A sample fragment is shown below.

```
    <Resource
        name="jdbc/auth"
        description="Sample authentication"
        type="javax.sql.DataSource"
        auth="Container"
        driverClassName="org.apache.derby.jdbc.ClientDriver"
        maxActive="10" maxIdle="3"
        maxWait="10000"
        password="PASSWORD"
        url="jdbc:derby://localhost:1527/authorize"
        validationQuery="values(1)"
        username="USER"/>
```

This Resource element describes a connection to the authorization database mentioned in the **Environments** section above. Some items to note about the configuration are:

- Replace the url, username, password, and driverClassName with those appropriate for the database being used
- validationQuery should be lightweight and return at least one row. It is database - specific so check your database documentation

Realm Element

The Realm element references the Resource element given above. The section of META-INF/context.xml describing the Realm element is given below.

```
    <Realm className="org.apache.catalina.realm.DataSourceRealm"
           userTable="APP.USERS"
           userNameCol="USERNAME"
           userCredCol="PASSWORD"
           userRoleTable="APP.AUTH"
           roleNameCol="ROLENAME"
           localDataSource="true"
           dataSourceName="jdbc/auth"/>
```

The Realm element above describes a table and column mapping between the database described in the **Environments** section and the required elements for authorization and authentication. Some items to note about the above configuration are listed below:

- . the realm is org.apache.catalina.realm.DataSourceRealm
- . dataSourceName must match the name given in the name attribute of the Resource element above
- . localDataSource="true" must be defined in order to use a Resource defined in META-INF/context.xml (the default is *false*)

Completed META-INF/context.xml

The completed META-INF/context.xml file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/HibernateApp">
    <Resource
        name="jdbc/auth"
        description="Sample authentication"
        type="javax.sql.DataSource"
        auth="Container"
        driverClassName="org.apache.derby.jdbc.ClientDriver"
        maxActive="10" maxIdle="3"
        maxWait="10000"
        password="PASSWORD"
        url="jdbc:derby://localhost:1527/authorize"
        validationQuery="values(1)"
        username="USER"/>
    <Realm className="org.apache.catalina.realm.DataSourceRealm"
           userTable="APP.USERS"
           userNameCol="USERNAME"
           userCredCol="PASSWORD"
           userRoleTable="APP.AUTH"
           roleNameCol="ROLENAME"
           localDataSource="true"
           dataSourceName="jdbc/auth"/>
</Context>
```

Summary for Everything in META-INF/context.xml

1. Add security constraints and information to WEB-INF/web.xml
2. Add Resource element to META-INF/context.xml
3. Add Realm element to META-INF/context.xml

Resource in $CATALINA_BASE/conf/server.xml and Realm in META-INF/context.xml

This configuration can be appropriate when multiple applications need to use the same authentication and authorization database. The JNDI resource is described in the GlobalNamingResources element of $CATALINA_BASE/conf/server.xml. Each application that requires authentication and authorization via this resource should a Realm definition in META-INF/context.xml referencing the global name.

Resource Element

The Resource element used in the GlobalNamingResources is the same one that is described above. The only difference is its placement. Below is the default GlobalNamingResources element (without comments) as shipped with Tomcat 6.

```
  <GlobalNamingResources>
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>
```

Adding the authentication and authorization resource to the above default implementation creates the following GlobalNamingResources element in $CATALINA_BASE/conf/server.xml.

```
  <GlobalNamingResources>
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
    <Resource
        name="jdbc/auth"
        description="Sample authentication"
        type="javax.sql.DataSource"
        auth="Container"
        driverClassName="org.apache.derby.jdbc.ClientDriver"
        maxActive="10" maxIdle="3"
        maxWait="10000"
        password="PASSWORD"
        url="jdbc:derby://localhost:1527/authorize"
        validationQuery="values(1)"
        username="USER"/>
  </GlobalNamingResources>
```

This entry makes the authentication and authorization database available to all applications by referencing the JNDI name jdbc/auth.

NOTE: In order to make the new Resource available, Tomcat will have to be restarted once the $CATALINA_BASE/conf/server.xml file has been modified.

Realm Element

Finally, in order for the web application to use this authentication and authorization resource, a Realm element needs to be added to META-INF/context.xml. An example is shown below.

```
    <Realm className="org.apache.catalina.realm.DataSourceRealm"
           userTable="APP.USERS"
           userNameCol="USERNAME"
           userCredCol="PASSWORD"
           userRoleTable="APP.AUTH"
           roleNameCol="ROLENAME"
           dataSourceName="jdbc/auth"/>
```

Items to note are listed below.

- **localDataSource="true"** is no longer present, since the Resource is no longer local.
- **dataSourceName** refers to the name of the Resource element in $CATALINA_BASE/conf/server.xml

Completed META-INF/context.xml

The completed META-INF/context.xml file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/HibernateApp">
    <Realm className="org.apache.catalina.realm.DataSourceRealm"
           userTable="APP.USERS"
           userNameCol="USERNAME"
           userCredCol="PASSWORD"
           userRoleTable="APP.AUTH"
           roleNameCol="ROLENAME"
           dataSourceName="jdbc/auth"/>
</Context>
```

Summary for GlobalNamingResources and META-INF/context.xml

1. Add security constraints and information to WEB-INF/web.xml
2. Modify $CATALINA_BASE/conf/server.xml
    a. Add the Resource sub-element to GlobalNamingResources
    b. Restart Tomcat to make the new Resource availale
3. Add the Realm element to **each** META-INF/context.xml that requires authentication and authorization

Resource and Realm in $CATALINA_BASE/conf/server.xml

Sometimes every sub-element under a particular element requires the same set of authentication and authorization resources. Rather than duplicating the configuration for multiple resources, it may make sense to place both the Resource and Realm in $CATALINA_BASE/conf/server.xml. Possible scenarios are listed below.

- Resource in GlobalNamingResources and Realm in a Host element
  - The Resource provides the authentication and authorization JNDI resource to all components
  - The Realm makes authentication and authorization information available to all web applications under the Host element
- Resource in GlobalNamingResources and Realm in an Engine element
  - The Resource provides the authentication and authorization JNDI resource to all components
  - The Realm makes authentication and authorization information available to all hosts and applications under the Engine

Each web application that wishes to make use of the $CATALINA_BASE/conf/server.xml - defined Realm must still obviously have security constraints configured in WEB-INF/web.xml.* *

Cascading Realms

Tomcat resolves multiple Realm definitions by using the most specific one for a given element. Examples are given below.

- Realm definition in the Engine element of $CATALINA_BASE/conf/server.xml
  - Would be overridden by a Realm definition in a Host sub-element of the Engine element
  - Would be overridden by a Realm definition in the META-INF/context.xml for a particular application
- Realm definition in the Host element of $CATALINA_BASE/conf/server.xml
  - Would be overridden by a Realm definition in the META-INF/context.xml for a particular application

CombinedRealm

One way to manage multiple Realms in $CATALINA_BASE/conf/server.xml is to use a CombinedRealm. The CombinedRealm provides a container for other Realms (sub-Realms). These Realms are **tried in the order configured**, until an authentication match is is made or all sub-Realms are tried.

Care should be taken that authentication and authorization information **does not unintentionally overlap**. Some of the consequences are discussed below.

- App1 uses Username/Password/Role from the first sub-Realm
- App2 uses Username/Password/Role from the second sub-Realm

If a username/password for App2 exists in the first sub-Realm, then authorization depends on whether or not the appropriate username/role also exists in the first sub-Realm.

If a role for App2 exists in the first sub-Realm, then a user authenticating in that sub-Realm could gain inappropriate access to App2 depending on the username/role mapping.

There are also benefits to this approach. One sub-Realm could be used as an "administrator" Realm, while other sub-Realms could provide authentication and authorization for specific applications.

Configuration Using CombinedRealm

The following steps can be used to configure a DataSource Realm in $CATALINA_BASE/conf/server.xml using a CombinedRealm.

Resource Element

Add the required Resource element to the GlobalNamingResources element in $CATALINA_BASE/conf/server.xml. Below is the default GlobalNamingResources element (without comments) as shipped with Tomcat 6.

```
<GlobalNamingResources>
  <Resource name="UserDatabase" auth="Container"
            type="org.apache.catalina.UserDatabase"
            description="User database that can be updated and saved"
            factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
            pathname="conf/tomcat-users.xml" />
</GlobalNamingResources>
```

Adding the authentication and authorization resource to the above default implementation creates the following GlobalNamingResources element in $CATALINA_BASE/conf/server.xml.

```
  <GlobalNamingResources>
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
    <Resource
        name="jdbc/auth"
        description="Sample authentication"
        type="javax.sql.DataSource"
        auth="Container"
        driverClassName="org.apache.derby.jdbc.ClientDriver"
        maxActive="10" maxIdle="3"
        maxWait="10000"
        password="PASSWORD"
        url="jdbc:derby://localhost:1527/authorize"
        validationQuery="values(1)"
        username="USER"/>
  </GlobalNamingResources>
```

This entry makes the authentication and authorization database available to all applications by referencing the JNDI name jdbc/auth.

NOTE: In order to make the new Resource available, Tomcat will have to be restarted once the $CATALINA_BASE/conf/server.xml file has been modified.

Realm Element

In order to avoid overriding the existing Engine-level Realm element in Tomcat's default $CATALINA_BASE/conf/server.xml, a CombinedRealm container will be used.

First, here is the default Realm as shipped with Tomcat 6.

```
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
       resourceName="UserDatabase"/>
```

Surround this Realm element with another Realm element defining the CombinedRealm. Within that element place both the default Tomcat UserDatabaseRealm and the DataSourceRealm. The resulting section of $CATALINA_BASE/conf/server.xml will look like the following.

```
<Realm className="org.apache.catalina.realm.CombinedRealm">
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
           resourceName="UserDatabase"/>
    <Realm className="org.apache.catalina.realm.DataSourceRealm"
           userTable="APP.USERS"
           userNameCol="USERNAME"
           userCredCol="PASSWORD"
           userRoleTable="APP.AUTH"
           roleNameCol="ROLENAME"
           dataSourceName="jdbc/auth"/>
</Realm>
```

NOTE: With both Realm and Resource information in $CATALINA_BASE/conf/server.xml, no Realm or Resource elements pertaining to authorization and authentication should appear in META-INF/context.xml. An application may require other Resource elements, but any Realm element in META-INF /context.xml will **override** that provided in $CATALINA_BASE/conf/server.xml.

Summary for Resource and Realm in $CATALINA_BASE/conf/server.xml

1. Add security constraints to the application's WEB-INF/web.xml
2. Add the JNDI resource to GlobalNamingResources in $CATALINA_BASE/conf/server.xml
3. Create a CombinedRealm at the appropriate level in $CATALINA_BASE/conf/server.xml (Engine is used in this example)
    a. Add the existing UserDatabaseRealm to the CombinedRealm as a sub-Realm
    b. Add the DataSourceRealm to the CombinedRealm as a sub-Realm
4. Restart Tomcat to read the configuration changes in $CATALINA_BASE/conf/server.xml

Summary

The following outline summarizes the three approaches discussed above.

1. Everything in META-INF/context.xml

    a. Add the Resource element describing the JNDI datasource
    b. Add the DataSourceRealm element
        i. add localDataSource="true" to reference the local JNDI datasource
2. Resource in $CATALINA_BASE/conf/server.xml and Realm in META-INF/context.xml
    a. Add the Resource element describing the JNDI datasource to GlobalNamingResources in $CATALINA_BASE/conf/server.xml
        i. Restart Tomcat to read the new Resource
    b. Add the DataSourceRealm element to the application's META-INF/context.xml
3. Resource and Realm in $CATALINA_BASE/conf/server.xml
    a. Add the Resource element describing the JNDI datasource to GlobalNamingResources in $CATALINA_BASE/conf/server.xml
    b. Add a CombinedRealm Realm element the the Engine element of $CATALINA_BASE/conf/server.xml
        i. Place the exisitng UserDatabaseRealm inside this CombinedRealm Realm element
        ii. Place the application's DataSourceRealm inside this CombinedRealm Realm element
    c. Restart Tomcat to read the new $CATALINA_BASE/conf/server.xml
    d. Make sure that no overriding Realms or Resources are present in the application's META-INF/context.xml file

---

CategoryFAQ